LLNL-CONF-461652

ClusteringWiki: Personalized and Collaborative Clusteringof Search Results

D. C. Anastasiu, D. J. Buttler, B. J. Gao

November 4, 2010

World Wide Web
Hyderabad, India
March 28, 2011 through April 1, 2011

ClusteringWiki: Personalized and Collaborative Clustering of Search Results

David C. Anastasiu
Texas State University-San Marcos
San Marcos, TX, USA
da1143@txstate.edu

David Buttler
Lawrence Livermore National Laboratory
Livermore, CA, USA
buttler1@llnl.gov

Byron J. Gao
Texas State University-San Marcos
San Marcos, TX, USA
bgao@txstate.edu

ABSTRACT

How to organize and present search results plays a critical role in the utility of search engines. Due to the unprecedented scale of the Web and diversity of search results, the common strategy of ranked lists has become increasingly inadequate, and clustering has been considered as a promising alternative. Clustering divides a long list of disparate search results into a few topic-coherent clusters, allowing the user to quickly locate relevant results by topic navigation. While many clustering algorithms have been proposed that innovate on the automatic clustering procedure, we introduce `ClusteringWiki`, the first prototype and framework for personalized clustering that allows direct user editing of the clustering results. Through a Wiki interface, the user can edit and annotate the membership, structure and labels of clusters for a personalized presentation. In addition, the edits and annotations can be shared among users as a mass-collaborative way of improving search result organization and search engine utility.

1. INTRODUCTION

The way search results are organized and presented has a direct and significant impact on the utility of search engines. The common strategy has been using a flat ranked list, which works fine for homogeneous search results.

However, queries are inherently ambiguous and search results are often diverse with multiple senses. With a list presentation, the results on different sub-topics of a query will be mixed together. The user has to sift through many irrelevant results to locate those relevant ones.

With the rapid growth in the scale of the Web, queries have become more ambiguous than ever. For example, there are more than 20 entries in Wikipedia for different renown individuals under the name of Jim Gray.

Consequently, the diversity of search results has increased to the point that we must consider alternative presentations, providing additional structure to flat lists so as to effectively minimize browsing effort and alleviate information overload [4, 10, 2]. Over the years clustering has been accepted as the most promising alternative.

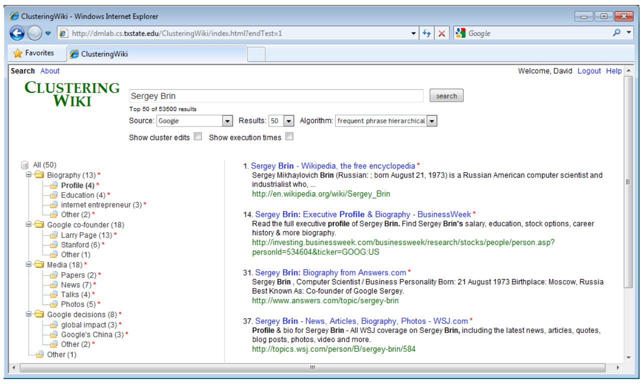
Clustering is the process of organizing objects into groups or clusters that exhibit internal cohesion and external isolation. Based on the common observation that it is much easier to scan a few topic-coherent groups than many individual documents, clustering can be used to categorize a long list of disparate search results into a few clusters such that each cluster represents a homogeneous sub-topic of the query. Meaningfully labeled, these clusters form a topic-wise non-predefined, faceted search interface, allowing the user to quickly locate relevant and interesting results. There is good evidence that clustering improves user experience and search result quality [7].

Given the significant potential benefits, search result clustering has received increasing attention in recent years from the communities of information retrieval, Web search and data mining [4, 10, 5, 9, 6]. In the industry, well-known commercial clustering search engines include Clusty (www.clusty.com), iBoogie (www.iboogie.com) and CarrotSearch (carrot-search.com).

Despite the high promise of the approach and a decade of endeavor, cluster-based search engines have not gained prominent popularity, evident by Clusty's Alexa rank. This is because clustering is known to be a hard problem, and search result clustering is particularly hard due to its high dimensionality, complex semantics and unique additional requirements beyond traditional clustering.

As emphasized in [9] and [2], the primary focus of search result clustering is NOT to produce optimal clusters, an objective that has been pursued for decades for traditional clustering with many successful automatic algorithms. Search result clustering is a highly user-centric task with *two unique additional requirements*. First, clusters must form interesting sub-topics or facets from the user's perspective. Second, clusters must be assigned informative, expressive, meaningful and concise labels. Automatic algorithms often fail to fulfill the human factors in the objectives of search result clustering, generating meaningless, awkward or nonsense cluster labels [2].

In this paper, we explore a completely different direction in tackling the problem of clustering search results, utilizing the power of direct user intervention and mass-collaboration. We introduce `ClusteringWiki`, the first prototype and framework for personalized clustering that allows direct *user* editing of the *clustering results*. This is in sharp contrast with existing approaches that innovate on the *automatic* algorithmic *clustering procedure*.


ClusteringWiki - Windows Internet Explorer
http://dmlab.cs.txstate.edu/ClusteringWiki/index.html?endTest=1
Google
Favorites
ClusteringWiki
Search
About
Welcome, David
Logout
Help
CLUSTERING WIKI
search

Sergey Brin

Top 50 of 53500 results

Source: Google Results: 50 Algorithm: frequent phrase hierarchical

Show cluster edits ☐ Show execution times ☐

- All (50)
  - Biography (13) *
    - **Profile (4)** *
    - Education (4) *
    - internet entrepreneur (3) *
    - Other (2) *
  - Google co-founder (18)
    - Larry Page (13) *
    - Stanford (6) *
    - Other (1)
  - Media (18) *
    - Papers (2) *
    - News (7) *
    - Talks (4) *
    - Photos (5) *
  - Google decisions (8) *
    - global impact (3) *
    - Google's China (3) *
    - Other (2) *
  - Other (1)

1. Sergey **Brin** - Wikipedia, the free encyclopedia *
Sergey Mikhaylovich **Brin** (Russian: ; born August 21, 1973) is a Russian American computer scientist and industrialist who, ...
http://en.wikipedia.org/wiki/Sergey_Brin

14. Sergey **Brin:** Executive **Profile** & Biography - BusinessWeek *
Read the full executive **profile** of Sergey **Brin.** Find Sergey **Brin's** salary, education, stock options, career history & more biography.
http://investing.businessweek.com/businessweek/research/stocks/people/person.asp?personId=534604&ticker=GOOG:US

31. Sergey **Brin:** Biography from Answers.com *
Sergey **Brin** , Computer Scientist / Business Personality Born: 21 August 1973 Birthplace: Moscow, Russia Best Known As: Co-founder of Google Sergey.
http://www.answers.com/topic/sergey-brin

37. Sergey **Brin** - News, Articles, Biography, Photos - WSJ.com *
Profile & bio for Sergey **Brin** - All WSJ coverage on Sergey **Brin,** including the latest news, articles, quotes, blog posts, photos, video and more.
http://topics.wsj.com/person/B/sergey-brin/584

Figure 1: Snapshot of `ClusteringWiki`.

In `ClusteringWiki`, the user can edit and annotate the membership, structure and labels of clusters through a Wiki interface to personalize her search result presentation. Such edits and annotations can be implicitly shared among users as a mass-collaborative way of improving search result organization and search engine utility. This approach is in the same spirit of the current trends in the Web, like Web 2.0, semantic web, personalization and mass collaboration.

Clustering algorithms fall into two categories: partitioning and hierarchical. Regarding clustering results, however, a hierarchical presentation generalizes a flat partition. Based on this observation, `ClusteringWiki` handles both clustering methods smoothly by providing editing facilities for cluster hierarchies and treating partitions as a special case. In practice, hierarchical methods are advantageous in clustering search results because they construct a topic hierarchy that allows the user to easily navigate search results at different levels of granularity.

Figure 1 shows a snapshot of `ClusteringWiki`. The left-hand *label panel* presents a hierarchy of cluster labels. The right-hand *result panel* presents search results for a chosen cluster label. A logged-in user can edit the current clusters by creating, deleting, modifying, moving or copying nodes in the cluster tree. Each edit will be validated against a set of predefined consistency constraints before being stored.

Designing and implementing `ClusteringWiki` pose non-trivial technical challenges. User edits represent user preferences or constraints that should be respected and enforced next time the same query is issued. Query processing is time-critical, thus efficiency must be given high priority in maintaining and enforcing user preferences. Moreover, complications also come from the dynamic nature of search results that constantly change over time.

Cluster editing takes user effort. It is essential that such user effort can be properly reused. `ClusteringWiki` considers two kinds of reuse scenarios, *preference transfer* and *preference sharing*. The former transfers user preferences from one query to similar ones, e.g., from "David J. Dewitt" to "David Dewitt". The latter aggregates and shares clustering preferences among users. Proper aggregation allows users to collaborate at a mass scale and "vote" for the best search result clustering presentation.

Contributions. (1) We introduce `ClusteringWiki`, the first framework for personalized clustering in the context of search result organization. (2) `ClusteringWiki` allows

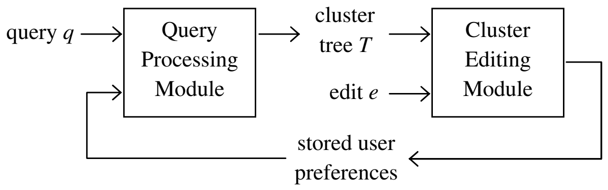


Figure 2: Main architecture of `ClusteringWiki`.

direct user editing of the clustering results through a Wiki interface, where user preferences are reused among similar queries as well as shared among users as a mass-collaborative way of improving search result organization and search engine utility. (3) We implement a prototype for `ClusteringWiki`, perform experimental evaluation and a user study, and maintain the prototype as a public Web service.

2. OVERVIEW

In this section, we overview the `ClusteringWiki` framework. Detailed descriptions of the framework and implementation can be found at [1]. As shown in Figure 2, the *query processing module* takes a query q and a set of stored user preferences as input to produce a cluster tree T that respects the preferences. The *cluster editing module* takes a cluster tree T and a user edit e as input to create/update a set of stored user preferences. Each user editing session usually involves a series of edits. The processing-editing cycle recurs over time.

Query processing. `ClusteringWiki` takes a query q from a user u and retrieves the search results R from a data source (e.g., Google). Then, it clusters R with a default clustering algorithm (e.g., frequent phrase hierarchical) to produce an initial cluster tree T_{init}. Then, it applies P, an applicable set of stored user preferences, to T_{init} and presents a modified cluster tree T that respects P.

Note that `ClusteringWiki` performs clustering. The modification should not alter R, the input data.

If the user u is logged-in, P will be set to $P_{q,u}$, a set of preferences for q previously specified by u. In case $P_{q,u} = \emptyset$, $P_{q',u}$ will be used on condition that q' is sufficiently close to q. If the user u is not logged-in, P will be set to $P_{q,U}$, a set of aggregated preferences for q previously specified by all users. In case $P_{q,U} = \emptyset$, $P_{q',U}$ will be used on condition that q' is sufficiently close to q.

In the cluster tree T, the internal nodes, i.e., non-leaf nodes, contain cluster labels and are presented on the left-hand *label panel*. Each label is a set of keywords. The leaf nodes contain search results, and the leaf nodes for a selected label are presented on the right-hand *result panel*. A search result can appear multiple times in T. The root of T represents the query q itself and is always labeled with *All*. When it is chosen, all search results will be presented on the result panel. Labels other than *All* represent the various, possibly overlapping, sub-topics of q. When there is no ambiguity, *internal node*, *label node*, *cluster label* and *label* are used interchangeably in the paper. Similarly, *leaf node*, *result node*, *search result* and *result* are used interchangeably.

The search results for a chosen label are presented in the result panel in their original order when retrieved from the source. Sibling cluster labels in the label panel are ordered by lexicographically comparing the lists of original ranks of

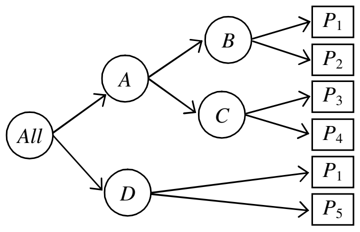


Figure 3: Example cluster tree.

their associated search results. For example, let A and D be two sibling labels as in Figure 3, where A contains P_1, P_2, P_3 and P_4 and D contains P_1 and P_5. Suppose that i in P_i indicates the original rank of P_i from the source. By comparing two lists $< 1, 2, 3, 4 >$ and $< 1, 5 >$, we put A in front of D. "Other" is a special label that is always listed at the end behind all its siblings.

Cluster editing. If logged-in, a user u can edit the cluster tree T for query q by creating, deleting, modifying, moving or copying nodes. User edits will be validated against a set C of consistency constraints before being written to $P_{q,u}$.

The set C contains predefined constraints that are specified on, for example, the size of clusters, the height of the tree and the length of labels. These constraints exist to maintain a favorable user interface for fast and intuitive navigation. The cluster tree T is *consistent* if it satisfies all the constraints in C.

In particular, `ClusteringWiki` implements the following categories of atomic *primitive* edits. With a series of such edits, a user can produce *any* consistent cluster tree.

- e_1: copy a label node to another non-bottom label node as its child. E.g., in Figure 3, we can copy D to A. We call a label node a *bottom label node* if it directly connects to search results.
- e_2: copy a result node to a bottom label node. E.g., in Figure 3, we can copy P_3 to D, but not to A, which is not a bottom label node.
- e_3: modify a non-root label node. E.g., in Figure 3, we can modify D to E.
- e_4: delete a non-root node, which can be either a label node or a result node. E.g., in Figure 3, we can delete P_5.
- e_5: create a label node, which can be either a non-bottom or bottom label node. In particular, recursive creation of non-bottom labels is a way to add levels to cluster trees. In Figure 3, we can add E as parent of D.

Note that, user editing can possibly generate *empty labels*, i.e., labels that do not contain any search results and thus not on any path. Such labels will be trimmed.

To add convenience, `ClusteringWiki` also implements several other types of edits. For example, move (instead of copy as in e_1) a label node to another non-bottom label node as its child, or move (instead of copy as in e_2) a result node to a bottom label node. Such a move edit can be considered as a copy edit followed by a delete edit.

By combining preferences in $P_{q,u}$ for all users who have edited the cluster tree T for query q, we obtain $P_{q,U}$, a set of aggregated preferences for query q. We use P_u to denote the collection of clustering preferences by user u for all queries, which is a set of sets of preferences such that $\forall q, P_{q,u} \in P_u$. We also use P_U to denote the collection of aggregated preferences by all users for all queries, which is a set of sets of aggregated preferences such that $\forall q, P_{q,U} \in P_U$. P_u and P_U are maintained over time and used by `ClusteringWiki` in processing queries for the user u.

Design principles. In a search result clustering engine, there are significant uncertainties from the data to the clustering algorithm. Wiki-facilitated personalization further adds substantial complications. Simplicity should be a key principle in designing such a complex system. `ClusteringWiki` adopts a simple yet powerful *path approach.*

With this approach, a cluster tree T is decomposed into a set of root-to-leaf *paths* that serve as independent editing components. A path always starts with *All* (root) and ends with some search result (leaf). In `ClusteringWiki`, maintenance, aggregation and enforcement of user preferences are based on simple path arithmetic. Moreover, the path approach is sufficiently powerful, being able to handle the finest user preference for a cluster tree.

In particular, each edit of T can be interpreted as operations on one or more paths. There are two primitive operations on a path p, *insertion* of p and *deletion* of p. A modification of p to p' is simply a deletion of p followed by an insertion of p'.

For each user u and each query q, `ClusteringWiki` maintains a set of paths $P_{q,u}$ representing the user edits from u for query q. Each path $p \in P_{q,u}$ can be either *positive* or *negative.* A positive path p represents an insertion of p, meaning that the user prefers to have p in T. A negative path $-p$ represents a deletion of p, meaning that the user prefers not to have p in T. Two *opposite* paths p and $-p$ will cancel each other out. The paths in $P_{q,u}$ may be added from multiple editing sessions at different times.

To aggregate user preferences for query q, `ClusteringWiki` first combines the paths in all $P_{q,u}$, $u \in U$, where U is the set of users who have edited the cluster tree of q. Then, certain statistically significant paths are selected and stored in $P_{q,U}$.

Suppose in processing query q, P is identified as the applicable set of paths to enforce. `ClusteringWiki` first combines the paths in P and the paths in T_{init}, where T_{init} is the initial cluster tree. Then, it presents the combined paths as a tree, which is the cluster tree T. The combination is straightforward. For each positive $p \in P$, if $p \notin T_{init}$, add p to T_{init}. For each negative $p \in P$, if $p \in T_{init}$, remove p from T_{init}.

Reproducibility. It is easy to verify that `ClusteringWiki` has the property of reproducing edited cluster trees. In particular, after a series of user edits on T_{init} to produce T, if T_{init} remains the same in a subsequent query, exactly the same T will be produced after enforcing the stored user preferences generated from the user edits on T_{init}.

3. DEMONSTRATION

`ClusteringWiki` [1] was implemented as an AJAX-enabled Java Enterprise Edition 1.5 application. The prototype is maintained on an average PC with Intel Pentium 4 3.4 GHz CPU and 4Gb RAM running Apache Tomcat 6.

We have conducted a comprehensive experimental evaluation on the correctness, efficiency, utility (by a paid user

[1]dmlab.cs.txstate.edu/ClusteringWiki.

study) and usability of ClusteringWiki. A complete report can be found at [1]. In the following, we focus on a guided demonstration of the services and functionality of the ClusteringWiki prototype.

Services and functionality. ClusteringWiki provides search service using multiple data sources, including Google AJAX Search API (code.google.com/apis/ajaxsearch), Yahoo! Search API (developer.yahoo.com/search/web/ webSearch.html), and local Lucene indexes built on top of the New York Times Annotated Corpus [8] and several datasets from the TIPSTER (disks 1-3) and TREC (disks 4-5) collections (www.nist.gov/tac/data/data_desc.html). The Google API can retrieve a maximum of 8 results per request and a total of 64 results per query. The Yahoo! API can retrieve a maximum of 100 results per request and a total of 1000 results. Due to user licence agreements, the New York Times, TIPSTER and TREC datasets are not available publicly.

ClusteringWiki facilitates four built-in clustering algorithms: k-means flat, k-means hierarchical, frequent phrase flat and frequent phrase hierarchical. The hierarchical algorithms recursively apply their flat counterparts in a top-down manner to large clusters. These algorithms build an initial cluster tree based on 50 to 500 search results (snippets and titles) retrieved from a data source.

The k-means algorithms follow a strategy that generates clusters before labels. They use a simple approach to generate cluster labels from titles of search results that are the closest to cluster centers. The parameter k is heuristically determined based on the size of the input.

The frequent phrase algorithms follow a strategy that generates labels before clusters. The labels are selected frequent phrases extracted from a suffix tree. A search result r belongs to a label L if r contains the keywords in L. These algorithms are able to generate very meaningful labels.

Based on the initial cluster tree, the main functionality provided by ClusteringWiki includes clustering personalization, clustering aggregation and preference transfer, to be described in the following.

Demonstration scenarios. After login, you are able to personalize a cluster tree for a query, e.g. "Sergey Brin". You can edit the tree labels in the label panel by renaming, creating, copying, moving and deleting labels. You can also re-assign the cluster membership of search results by copy, move and delete operations.

These edits must conform to a predefined set of consistency constraints. To reduce user editing effort, cluster edits in ClusteringWiki are available through context menus attached to labels and results, and only pre-validated operations will be made available. You can drag and drop a result or a label in addition to cutting and pasting to perform a move operation. Edited labels and results are marked by a red asterisk.

To verify that ClusteringWiki retains personal preferences, you can log out, log in, and issue the same query. You should see your previous edits have been incorporated into the cluster tree. This personalized cluster tree should allow you to explore the search results more effectively.

To demonstrate aggregated clustering, ClusteringWiki lists the top 10 queries that have been edited by the most users. You (without login) can choose some of these queries and observe how the aggregated trees differ from the initial trees. While they may not contain direct personal edits, these aggregated trees should reflect the collaborative effort and common preferences of many users. You should benefit from the "voted" tree without any cost of editing effort. To test how ClusteringWiki behaves in aggregating edits, you can also create multiple accounts, log in, issue the same query, edit the cluster trees using each account, and then log out and examine the aggregated tree.

Cluster editing takes user effort. ClusteringWiki attempts to reuse such effort as much as possible. While preference aggregation can be considered sharing among users, preference transfer is sharing among queries. In ClusteringWiki, preference transfer is executed regardless of your login status. When logged in, your personal preferences of a similar query are transferred. When logged out, the aggregated preferences are transferred. For example, you can issue a query "Sergey Brin" and edit the cluster tree. Then you can issue a similar query "Sergey M. Brin" and observe how those stored preferences for "Sergey Brin" are enforced in producing the cluster tree for "Sergey M. Brin".

4. CONCLUSION

There are many interesting directions for future work, from fundamental semantics and functionalities of the framework to convenience features, user interface and scalability. One particular interesting direction is to seamlessly integrate the Wiki-style personalization of search results [3] with the personalization of search result clusters, providing a more complete solution for personalized and collaborative information retrieval and Web search.

5. ACKNOWLEDGMENTS

This work (LLNL-CONF-461652) was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

6. REFERENCES

[1] Clusteringwiki technical report. *dmlab.cs.txstate.edu/ClusteringWiki/pdf/cw.pdf*, 2010.

[2] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):1–38, 2009.

[3] B. J. Gao and J. Jan. Rants: a framework for rank editing and sharing in web search. In *WWW*, 2010.

[4] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR*, 1996.

[5] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW*, 2004.

[6] J. Lee, S.-w. Hwang, Z. Nie, and J.-R. Wen. Query result clustering for object-level search. In *KDD*, 2009.

[7] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[8] E. Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 2008.

[9] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, 2007.

[10] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. In *WWW*, 1999.